

31  
150pts.

# ARUN

Enciclopedia Práctica del Spectrum



Nueva Lente/Ingelek





# EL MICRODRIVE



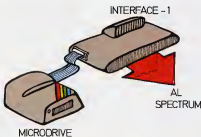
El ZX MICRODRIVE es el sistema de almacenamiento de datos de acceso rápido de mayor difusión en el mercado. Ello es debido a dos razones fundamentales: su bajo costo en relación a su capacidad y prestaciones, y su facilidad de manejo. Para poder conectar a nuestro Spectrum una unidad MICRODRIVE es preciso adquirir el correspondiente interfaz denominado ZX INTERFACE 1. Este interfaz, uno de los dos comercializados por Sinclair Research Ltd., podemos denominarlo como de «gestión» puesto que, además de permitir aceptar al ordenador hasta 8 unidades MICRODRIVE, soporta la conexión a impresoras y otros periféricos del estándar RS232, así como el establecimiento de una red (NET) de comunicación de hasta 64 Spectrums.

El MICRODRIVE permite los mismos tipos de almacenamiento de datos del casete: programas, bloques de memoria, pantallas y matrices, pero incluye además una nueva forma denominada archivo secuencial. Asimismo incorpora la posibilidad de ejecución automática de un programa determinado, la de proteger programas, y la de hacer copias de ficheros.

El MICRODRIVE permite almacenar programas, bloques de memoria, pantallas, matrices y otros archivos secuenciales.

## MICRODRIVE

- PROGRAMAS
- PANTALLAS
- BLOQUES DE MEMORIA
- MATRICES
- ARCHIVOS SECUENCIALES



Para poder conectar a nuestro Spectrum una unidad MICRODRIVE, es necesario adquirir un INTERFACE 1.

## FORMAT

Para que un cartucho de MICRODRIVE virgen sea utilizable es preciso someterlo a la operación de formateado. Esta operación consiste en la explotación de la cinta sin fin que constituye el cartucho con dos objetivos fundamentales: dividirlo en sectores, los cuales albergarán bloques de datos, y realizar al mismo tiempo la comprobación de que estas porciones son aptas para las operaciones de lectura/escritura, siendo eliminadas en caso contrario.

Es por este motivo que el espacio disponible en el cartucho, al concluir la operación de formateado, no es siempre el mismo. En condiciones normales este espacio debe oscilar entre los 85 y 95 Kbytes. De no ser así, es síntoma de detención del

! !

Para que un cartucho de MICRODRIVE sea utilizable es preciso formatearlo. Esta operación consiste en desplazar el cartucho para dividirlo en sectores, y realizar la comprobación de que son aptos para las operaciones de lectura/escritura.



Una sesión de data rano de un cartucho es el número de carga de carga y grabación



El INTERFACE 1 permite aceptar hasta 8 unidades MICRODRIVE, impresoras y otros periféricos del estándar RS232 así como el establecimiento de una red (NET) de comunicación de hasta 64 Spectrum.



El MICRODRIVE permite almacenar en un cartucho programas, bloques de memoria, películas, matrices de datos y archivos de cuadros.

cartucho, lo cual debe hacernos pensar en repetir en un futuro no muy lejano en evitación de malos mayores.

El ordenador se toma unos 30 segundos en realizar esta operación. Durante este tiempo, el conector primero parpadea y más tarde se ilumina verticalmente por unas bandas anchas de color oscuro, presentando finalmente el mensaje O.K. Debemos tener en cuenta que cuando se realiza la operación de formateado, la información anterior contenida en el cartucho se pierde. Esta contingencia, que debería representarse una ventaja por permitir borrar todos los contenidos de un cartucho rápidamente, nos obliga a proceder con extremo cuidado en no someter a esta operación a un cartucho inadecuado.

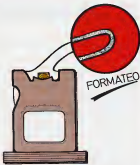
El formato general de la instrucción es:

**FORMAT "m";j;"nombre"**

Donde m indica que se trata de formatear un cartucho en una unidad MICRODRIVE de número m, cuyo valor puede oscilar entre 1 y 8.

El primer parámetro es fijo para todas las operaciones con MICRODRIVE, pudiendo escribirse en mayúsculas o minúsculas indistintamente. El segundo parámetro debe corresponder al número

de la sesión del dispositivo de los cartuchos es el elemento considerable en el tiempo de carga y grabación.



Para que un cartucho de MICRODRIVE sea utilizable en un sistema Spectrum, es decir, explorar la cinta de la unidad en la que deseamos efectuar la operación.

de la unidad en la que deseamos efectuar la operación. Caso de existir una única unidad debe especificarse siempre un 1, de existir más, irán conectadas una a la otra hacia la izquierda de la primera, siendo su numeración correlativa desde 2 hasta 8. El tercer parámetro contendrá el nombre (1 a 10 caracteres) que deseamos dar al cartucho.

Los tres parámetros referenciados son de uso obligatorio. En caso de olvidarnos del primero o de especificarlo incorrectamente, obtendremos el mensaje de error Invalid device expansion (identificación de periférico incorrecta). Asimismo, si especificamos como segundo parámetro el número 0 o uno superior a 8 obtendremos el mensaje Invalid drive number (número de MICRODRIVE incorrecto). Por último, si señalamos un nombre no válido para el cartucho (una cadena vacía o excediendo 10 caracteres), aparecerá el mensaje Invalid name (nombre incorrecto).

Las sucesivas operaciones de borrado y escritura de archivos contenidas en los cartuchos va deteriorando éstos de forma progresiva. Un síntoma de que este hecho se está produciendo es el aumento considerable en el tiempo de carga y grabación. Una solución para evitar este problema es copiar el contenido del cartucho deteriorado sobre un segundo, realizando posteriormente un nuevo formateado del primero. Los cartuchos así

regeneradas, cobran nueva vida, obteniendo un mayor rendimiento en lo sucesivo hasta que vuelven a deteriorarse por el uso.

CAT

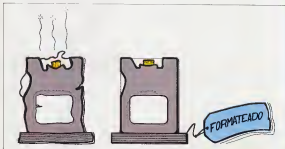
La sentencia CAT permite obtener un listado por pantalla, ordenado alfabéticamente, del contenido de un cartucho. Su formato general es:

CAT n

Donde n es obligatorio y debe ser un número de 1 a 8 especificando el número de unidad MICRODRIVE en la cual trabajar. La especificación incorrecta de este número provoca el error *Invalid number*.

CAT suele emplearse inmediatamente después de efectuar un FORMAT, para cerciorarnos de que la capacidad del cartucho es la adecuada y su nombre el correcto.

*Si deseamos recuperar un cartucho basta con transferir la información a otro cartucho, y volver a formatear el mismo.*



*La sentencia CAT permite obtener un listado por pantalla, ordenado alfabéticamente, del contenido de un cartucho.*

SAVE \*

La sentencia SAVE \* es la equivalente en MICRODRIVE de la sentencia SAVE para el casete, permitiendo almacenar en cartucho programas, bloques de memoria, pantallas y matrices de datos. El formato general de la sentencia es:

SAVE "\*"m";c;"nombre"



La sentencia CAT permite obtener un listado por pantalla, ordenado alfabéticamente, del contenido de un cartucho.



El MICRODRIVE es el sistema de almacenamiento de datos de más reciente desarrollo de mayor capacidad y prestaciones.



Para poder conectar a nuestro Spectrum una unidad MICRODRIVE es preciso adquirir el ZX INTERFACE 1.

## MICRODRIVE

SAVE \*



## CASETE

SAVE

La sentencia SAVE \* es el equivalente en MICRODRIVE de la sentencia SAVE para casete.

!!

Al realizar la operación de formateo, la información anterior contenida en el cartucho se pierde.

\*

El MICRODRIVE permite la inserción de los nombres de programas y datos en el directorio (CAT) del MICRODRIVE, aña diendo a estos el prefijo CHR# 0.

\*

En condiciones normales, la capacidad de un cartucho oscila entre los 65 y 95 Kbytes. De no ser así, es síntoma de deterioro del cartucho.

Dicho formato permite almacenar programas. Para conseguir la autoejecución, al igual que sucede con la sentencia SAVE, es necesario especificar la palabra clave LINE, seguida del número de línea a partir de la cual se desea que comience la ejecución del programa, al cargarlo la próxima vez en memoria:

SAVE "m";n;"nombre" LINE l

Donde l representa el número de línea desde el que se desea comience la ejecución.

En la autoejecución, cosa que no ocurre con el casete, no está permitida la operación de MERGE \* del programa así grabado en cartucho. En cierto modo, esta particularidad nos brinda un método sencillo de efectuar la protección de programas almacenados en cartucho contra la obtención de copias ilegales.

De forma similar a lo que sucede con el casete, para almacenar bloques de memoria en cartucho es necesario especificar la palabra clave CODE, seguida de dos valores numéricos que representan la dirección de comienzo y la longitud del bloque de bytes a grabar:

SAVE "m";n;"nombre" CODE d, l

Donde d representa la dirección de comienzo y l el número de bytes a transferir al cartucho. Para el caso concreto de las pantallas, basta con especificar la palabra clave SCREEN#:

SAVE "m";n;"nombre" SCREEN#

Por último, para almacenar el contenido de matrices de datos numéricos o de cadena, es necesario incluir la palabra clave DATA, seguida del nombre de la matriz y una apertura y cierre de paréntesis para indicar la naturaleza de los datos:

SAVE "m";n;"nombre" DATA x( )

SAVE "m";n;"nombre" DATA x\$( )

Donde x y x\$ representan los nombres de una matriz cualquiera, numérica o de cadena respectivamente.

Al igual que sucede con la sentencia FORMAT, es obligatorio especificar todos los parámetros, produciéndose en caso contrario alguno de los correspondientes mensajes de error.

## VERIFY \*

La sentencia VERIFY \* es también equivalente en MICRODRIVE a la sentencia VERIFY del ca

En la autoejecución se está permitida la operación de MERGE \* del programa en cartucho así grabado.





La sentencia VERIFY \* tiene por objeto daros la seguridad de que el almacenamiento en cartucho se ha realizado sin problemas.

esto, siendo aplicable a programas, bloques de memoria, pantallas y matrices de datos almacenados previamente en cartucho por medio de SAVE \*.

La verificación tiene por objeto daros la seguridad de que el almacenamiento en cartucho se ha realizado sin problemas. Por ello, y dada la alta velocidad a que el MICRODRIVE permite realizar esta operación, es siempre conveniente pasar por este punto inmediatamente después de cada SAVE \*. Los formatos que esta sentencia puede adoptar son, respectivamente:

```
VERIFY ***m**n:**nombre**
VERIFY ***m**n:**nombre** CODE d
VERIFY ***m**n:**nombre** SCREEN$
VERIFY ***m**n:**nombre** DATA xj )
VERIFY ***m**n:**nombre** DATA x$ )
```

Donde los parámetros deben seguir las reglas anteriormente comentadas.

A diferencia de lo que sucede con el casete, la sentencia VERIFY \* no comprueba byte a byte el contenido de la memoria del ordenador con el del cartucho, sino que se limita a constatar que el nombre solicitado se encuentra grabado y que sus diferentes bloques de datos se hallan enlazados correctamente.

En el MICRODRIVE, a diferencia de en el casete, si es posible la verificación de pantallas.

## LOAD \*

La sentencia LOAD \* es también equivalente en MICRODRIVE a la sentencia LOAD del casete, permitiendo la lectura de programas, bloques de memoria, pantallas y matrices de datos almacenados con SAVE \*. Los formatos que esta sentencia puede adoptar son, respectivamente:

```
LOAD ***m**n:**nombre**
LOAD ***m**n:**nombre** CODE d
LOAD ***m**n:**nombre** SCREEN$
LOAD ***m**n:**nombre** DATA xj )
LOAD ***m**n:**nombre** DATA x$ )
```

Donde los parámetros deben seguir las reglas anteriormente comentadas. Dado que el nombre del

No se permite en la verificación el la carga especificada una cadena vacía



!!

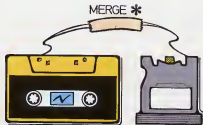
En la autoejecución, no está permitido el MERGE \* del programa en cartucho así grabado.

\*

Las sucesivas operaciones de borrado y escritura de archivos contenidos en los cartuchos van determinando el estado de forma progresiva.

\*

La sentencia SAVE \* permite almacenar en cartucho programas, bloques de memoria, pantallas y matrices de datos.



*La orden MERGE \* permite la fusión en memoria del programa contenido en cinta, con el contenido de cartucho.*

todos automáticamente a partir de la línea especificada al concluir el proceso de carga.

Por otro parte, en la carga de bloques de memoria debemos recordar que, en el caso de no especificar ningún parámetro, ésta se efectuará del total de los bytes contenidos en el bloque y a partir de la misma dirección de la que fueron grabados.

Sin embargo, podemos especificar el primer parámetro d. En este caso se cargará en memoria la totalidad del bloque de bytes, pero a partir de la nueva dirección especificada.

Por último, diremos que no podemos especificar un segundo parámetro l, determinando el número exacto de bytes que deseamos cargar como sucede con las lecturas de cinta. En el caso del MICRODRIVE obtendremos el error CODE amar, tanto por especificar menor como mayor número de bytes de los contenidos en el cartucho. En cualquier caso, en la carga desde cinta con un parámetro l menor que el original, se efectúa la operación pero con la obtención del error Tape loading error.

---

## MERGE \*

---

archivo es siempre imprescindible, no se permiten ni la verificación ni la carga especificando una opción vacía (equivalente a VERIFY "" y LOAD "").

Recordemos que los programas almacenados en cartucho con el parámetro LINE x, serán ejec:

La sentencia MERGE \* es equivalente en MICRODRIVE al MERGE de casete, permitiendo la fusión en memoria del programa contenido en ésta con uno procedente de cartucho.

En esta operación se tiene en cuenta tanto el área de programa propiamente dicha (líneas que lo

!

La verificación tiene por objeto darnos la seguridad de que el almacenamiento en cartucho se ha realizado sin problemas.

\*

VERIFY \* no comprueba byte a byte el contenido de la memoria del ordenador con el del cartucho sino que se limita a constatar que el nombre señalado se encuentra grabado y que sus diferentes bloques de datos se hallan enlazados consecutivamente.

*El MICRODRIVE permite la carga automática de un programa contenido en cartucho con el nombre de run.*





comparan), como la correspondiente zona de variables, de forma que los nuevos números de línea reemplazan a los anteriores y son incluidos en el programa si no existen, aplicándose esta misma regla al contenido de las variables. El formato general de la sentencia es:

**MERGE "m";n:"nombre"**

Recordemos aquí que los programas almacenados en cartucho por medio de la sentencia **SAVE \* LINE** no admiten la opción de **MERGE**, permitiendo únicamente la de **LOAD \*** (carga directa con autoejecución).

**ERASE**

Hasta ahora hemos tratado sobre los equivalentes a las funciones de grabación, verificación y lectura de programas, bloques de memoria, pantallas y matrices de datos, siendo característica general el empleo del mismo comando **BASIC** que para las operaciones en cinta, seguido de un asterisco (\*).

*La sentencia ERASE permite borrar físicamente un nombre contenido en cartucho.*



*El MICRODRIVE no permite nombres duplicados y avisa con un mensaje de error cuando se trata de grabar con un nombre ya existente.*

Sin embargo, al igual que **FORMAT** y **CAT** son sentencias específicas de este periférico, **ERASE** también lo es. La sentencia **ERASE** permite borrar físicamente un nombre contenido en el cartucho, de forma que el espacio ocupado por el mismo se suma al total del disponible en la unidad. El formato general de la sentencia es:

**ERASE "m";n:"nombre"**

Donde no es necesario especificar la clase de dato de que se trata: programa, bloque de memoria, pantalla o matriz de datos. La sentencia **ERASE** sirve para eliminar los contenidos innecesarios en el cartucho, pero es también de uso obligado en las sucesivas actualizaciones de los contenidos.

Si, por ejemplo, hemos cargado un programa en memoria para modificarlo, hemos de efectuar un **ERASE** de la copia almacenada en cartucho antes de ejecutar el **SAVE \*** de la nueva versión del programa. Esto se debe a que el **MICRODRIVE** no permite nombres duplicados, y avisa con un mensaje de error cuando se trata de grabar sobre un nombre existente. Writing to a "read" file.



Los programas almacenados en cartucho con el parámetro **LINE** son ejecutados automáticamente a partir de la línea especificada al construir el proceso de carga.



La sentencia **VERIFY \*** es aplicable a programas, bloques de memoria guardados y matrices de datos al almacenarse físicamente en cartucho por medio de **SAVE \***.

**!**

## CARGA AUTOMÁTICA

La unidad MICRODRIVE permite el proceso de carga automática de un programa contenido en un cartucho cuyo nombre debe ser **run**.

\*

La sentencia **ERASE** borrar fácilmente un nombre contenido en el cartucho, de forma que el espacio ocupado por el mismo se vaya al total del disponible en la unidad.

\*

La sentencia **MERGE** \* permite la fusión en memoria del programa contenido en caso con un procedimiento de cartucho.

\*

La sentencia **LOAD** \* permite la lectura de programas, bloques de memoria, pánfidos y matrices de datos almacenados con **SAVE** \*

La unidad MICRODRIVE permite el proceso de carga automática de un programa contenido en cartucho, cuyo nombre debe ser **run**. Esta función sólo es accesible cuando se enciende por primera vez el ordenador, o bien después de ejecutar un comando **NEW** o la inicialización del Sistema mediante un **USR 0**.

Para invocar esta función basta con escribir, después de cumplirse alguna de las condiciones antes expuestas, la palabra clave **RUN** seguida de **ENTER**. Con esta operación, el ordenador busca y carga en memoria el programa contenido en el cartucho del MICRODRIVE 1 cuyo nombre es **run**. Si el programa ha sido grabado con la opción de autoejecución, el caso más frecuente, éste se ejecutará automáticamente a partir de la línea especificada.

Esta opción permite, la creación de un programa que sirva de Menú de acceso a los comandos del cartucho, con la ventaja de cargarse de forma automática sin requerir conocimientos del BASIC SINCLAIR. Una muestra de su utilidad puede encontrarse en el propio cartucho de demostración suministrado con cada unidad MICRODRIVE.

El MICRODRIVE nos brinda como sistema de protección la invisibilidad de los nombres de programas en el directorio (CAT), con el prefijo **CHR\$ 0**.

**CHR\$ 0**

DOS  
ARCHIVO  
BB1  
CEAC  
DELTA  
29  
Ø OK, Ø 1

La carga automática de un programa por el microdrive sirve para la creación de menús de acceso a los contenidos del cartucho.

Para poder crear un programa de este tipo basta con confeccionar un programa BASIC, cumpliendo las condiciones deseadas, y grabarlo en cartucho con el nombre **run** (debe escribirse en minúsculas) y autoejecución a la línea de su comienzo.

## PROTECCION DE PROGRAMAS

Además de la prohibición del **MERGE** \* de los programas grabados con el parámetro **LINE**, el MICRODRIVE nos brinda otro sistema de protección, consistente en la invisibilidad de los nombres de programas y datos en el directorio (CAT) del MICRODRIVE.

Para conseguir este efecto basta con anteponer al nombre del programa, bloque de memoria pantalla o matriz de datos, el prefijo **CHR\$ 0** antes de la grabación, con o sin **LINE**, de la forma

**SAVE "m"m;CHR\$ 0+"nombre" LINE x**

Conviene tomar la precaución de anotar los nombres de los programas así grabados, dado que el no aparecer en el directorio del cartucho no podemos recuperarlos si olvidamos su nombre.



# PROCESADORES DE IDEAS



A polémica suscitada en torno a la inteligencia artificial en ordenadores, no insinúa una solución que unifique criterios sobre el tema. Muchos consideran procesos tales como la robótica o la síntesis de voz dentro de este campo, pues de alguna manera en ellos las máquinas son capaces de reproducir, más o menos fielmente, ciertas facetas del comportamiento humano, aunque ejecutan un trabajo meramente mecánico o poco creativo.

Otros no dudan en señalar que las actividades en las cuales la máquina sea capaz de realizar labores típicas de inferencia y relación de ideas, a fin de cuentas que demuestre por sí misma una capacidad para analizar sus «pensamientos», son las únicas válidas para ser consideradas dentro de este campo.

Pero, de una u otra manera, los avances realizados durante los últimos años pueden considerarse como un éxito, y no olvidemos que lejos de la utópica creencia por la cual los más sensacionalistas vaticinan un mundo futuro dominado por máquinas, no por hombres, los progresos realizados dentro de la inteligencia artificial están al servicio de todos. Demos, pues, un repaso al estado actual de esta área de la informática en constante desarrollo.

## UN BANCO DE PRUEBAS

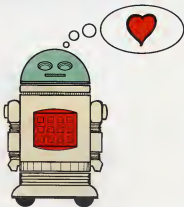
Resultado increíble como algunos de los grandes precursores de la inteligencia artificial, como Turing o Shannon, dedicaron parte de su trabajo a desarrollar juegos inteligentes o que al menos pareciera que lo son. Es más, en los orígenes de la inteligencia artificial, fueron muchos los que aseguraban, tan rotundo como equivocadamente, que pronto una máquina sería capaz de vencer al campeón del mundo de ajedrez. Por supuesto, incluso en nuestros días, el mejor de estos programas no tiene ninguna posibilidad, no ya contra

un gran maestro, sino contra un jugador de tipo medio.

Entonces, si los intentos en este sentido han constituido un sonoro fracaso, ¿por qué se sigue investigando? Un programa para jugar al ajedrez, por ejemplo, con toda perfección y garantía de éxito, debería implementarse desde el primer movimiento toda la gama posible de jugadas y sus variantes conforme avanza la partida; debería tenerlo previsto todo.

De hecho, el problema técnicamente no estriba en la construcción del programa, difícil de por sí, sino en el elevado tiempo de proceso necesario para concluir una partida. Supongo que nada es-

*Una de las mayores deficiencias de inteligencia artificial, sostiene que ha de tener capacidad de análisis de pensamiento por sí misma.*



!!

Los sistemas expertos se encargan de almacenar los conocimientos de las personas, especializados en lo más específico.

\*

En caso de ordenar que analice las distintas concepciones sobre inteligencia artificial, lo cual ha provocado una gran polémica.

\*

Los juegos inteligentes constituyen un excelente banco de pruebas previo a la evolución de otros juegos más serios.



En los juegos de la inteligencia artificial se asegura que una máquina sea capaz de vencer al campeón del mundo de ajedrez.

tará dispuesto a pasar docenas de años frente a una máquina esperando su genial movimiento. A pesar de la velocidad de proceso en los grandes superordenadores, se estima que el tiempo necesario para procesar una jugada hasta sus últimas consecuencias sería de miles de años.

Los juegos inteligentes constituyen un gran banco de pruebas previo a aplicaciones concretas.



Para evitarlo, se establecen ciertas simplificaciones, y es aquí donde radica la importancia de los juegos inteligentes: constituyen un excelente banco de pruebas previo a la ejecución de aplicaciones más serias. Por ejemplo, se desarrollan funciones de evaluación, donde se incluyen el valor técnico de cada pieza o la importancia relativa del dominio sobre ciertas casillas del tablero, es decir, el ordenador es capaz de identificar como movimiento adecuado el cambiar un alfil enemigo por su dama. Por tanto, tiene una cierta «conciencia» y un funebre de ella es capaz de tomar decisiones.

Pero claro, esto no basta. Además se incorporan al programa principios heurísticos, los cuales facilitan a la máquina para conseguir su objetivo ejecutando un movimiento, utilizando técnicas basadas más en la experiencia que en el cálculo. Desde luego, de esta manera no se garantiza que la decisión adoptada por el ordenador sea la mejor, pero sí se asemeja la forma de jugar de la máquina a como lo haría un humano, imitando su capacidad de improvisación.

## SISTEMAS EXPERTOS

El estudio de las funciones de evaluación y la inclusión de principios heurísticos, ha conducido a otro logro dentro de la inteligencia artificial: los sistemas expertos. Estos se encargan de almacenar los conocimientos de las personas especializadas en determinados temas específicos, con el objeto de facilitar soluciones en caso de ausencia del especialista.

Resuelven difíciles problemas, tan bien o mejor que los expertos humanos, razonan las decisiones que toman, consideran simultáneamente, distintos hipótesis interactivas entre sí, son capaces de dialogar en lenguaje natural, admiten descripciones simbólicas, y desde luego, justifican detalladamente sus conclusiones.

Pero lo más importante es la capacidad para, al igual que ocurre con los conocimientos adquiridos por la experiencia de una persona, manejar datos ambiguos e incluso erróneos. Es decir, simulan la forma de pensar y razonar de un experto humano, ya que estos, muchas veces, son incapaces de explicar el porqué las cosas son así. Pero están seguros de que lo son.

Sus aplicaciones son obvias. Por ejemplo, un sistema de este tipo aplicado a la medicina, solicita del usuario los síntomas que presenta el paciente, y a partir de ellos, facilita el diagnóstico y con

¿la terapia posible a seguir, él, sin duda, so-  
lució todo si no se encuentra un médico cerca.  
O en el control de la seguridad de las grandes  
plantas de producción de energía nuclear. En un  
determinado momento, el operario encargado de  
verificar el correcto funcionamiento de la control,  
puede verse saturado ante la avalancha de posi-  
bles influencias causales, las cuales conducen  
inevitablemente al accidente y posteriormente, a  
la catástrofe. Mantener la «cabeza fría» puede ser  
de vital importancia, y un sistema experto indica  
al rápidamente, a la par de conservar los «re-  
vices» en calma, las acciones a tomar para solu-  
cionar el problema, evitando que ésta vaya a ma-  
yores.

No obstante, sus aplicaciones siguen siendo uti-  
lizadas de forma puramente consultiva, dejando  
la decisión final a tomar en manos de los huma-  
nos. Hemos de tener en cuenta que son incapaces  
de llegar a conclusiones intuitivamente, y a sus  
razonamientos no se basan, como en el caso huma-  
no, en el sentido común.



## LA QUINTA GENERACION

Bajo esta denominación se conoce un ambicioso  
proyecto, emprendido en Japón, encaminado a  
convertir en realidad la tecnología de la inteli-  
gencia en ordenador. El programa cuenta con un  
presupuesto de 500 millones de dólares y una  
duración de diez años.

Para ello, el gobierno japonés ha reunido a un  
grupo de jóvenes «cerebrós» dentro del campo de  
la informática, y en particular, de la inteligencia  
artificial, ninguno de los cuales en el momento de  
iniciarse el proyecto contaba con más de treinta  
y cinco años.

La principal pretensión es la de construir orde-  
nadores de arquitectura distinta de la actual (de-  
nominado de Von Neumann), capaces de revelar  
construir la filosofía existente sobre estas máquinas  
hasta el momento. Es decir, se pasará a la tec-  
nología del «qué» debe hacer un ordenador (bas-  
tará con decirlo), contrariamente a la de nues-  
tros días, consistente en indicarle, paso a paso,  
«cómo» debe llevar a cabo su trabajo.

Durante la reciente exposición internacional  
TSUKUBA '85, celebrada en Japón, brillaron por  
su ausencia los caparados amigos dentro de este  
campo. En este sentido, las opiniones son para to-  
dos los gustos: unos opinan que es todavía pronto  
para desvelar los logros conseguidos, y por ello

*Los sistemas expertos se encargan de abstractar los  
razonamientos de los especialistas.*

se mantienen en secreto, mientras otros indican  
que la realidad es muy distinta, no habiéndose al-  
canzado por el momento ningún resultado espe-  
cialmente significativo.

Pero si es cierto, no obstante, la fuerte inversión

*Isaac Asimov en su ficticio manual de robótica, enseñó  
una serie de leyes básicas que siempre habrán que tener en  
cuenta.*



**!**

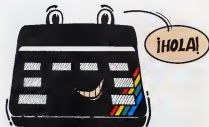
Parte del trabajo desa-  
rrollado en el campo  
de la inteligencia arti-  
ficial se ha dedicado a  
los juegos inteligentes.

\*

Para la escritura de  
programas inteligentes  
se han diseñado  
especialmente los  
lenguajes como ALGOL,  
LISP y PROLOG.

\*

Los sistemas de bases  
datos de voz por par-  
te de un ordenador es  
uno de los campos  
más estudiados en es-  
tos últimos años.



AL VILLO COMANDO convertido el Spectrum reconoce el sonido de las palabras que se aprende con entonación.

efectuado, tanto por los Estados Unidos, como por Japón en el proyecto y sea cual sea el estado actual de los cosas, el beneficio es palpable, pues se está contribuyendo de forma espectacular al rápido desarrollo de la inteligencia artificial.

El programa japonés sobre inteligencia artificial «La Quinta Generación» cuenta con un presupuesto de 500 millones de dólares.

## LENGUAJES

La escritura de programas inteligentes puede efectuarse empleando muchos de los lenguajes existentes en la actualidad, pero al igual que ocurre con otras aplicaciones, algunos han sido específicamente diseñados con este propósito. Entre los más populares cabe señalar el ALGOL (ALGORITMO Language, lenguaje algorítmico), de estructura similar al FORTRAN (FORMULA TRANSLATOR, traductor de fórmulas) pero mucho más flexible que éste. Su origen data de principios de los sesenta, aunque posteriormente se han implementado versiones mejoradas. Al objeto de facilitar el manejo de información simbólica y grandes listas de datos, surgió en los sesenta un lenguaje denominado LISP (LIST Processor, procesador de listas), el cual pronto se popularizó por sus aplicaciones dentro de la investigación para la inteligencia artificial. Un programa escrito en LISP es una serie de listas de funciones matemáticas, las cuales operan sobre otras listas de datos, que cambian, frecuentemente, durante el proceso.

Finalmente, comentar un lenguaje el cual, según todos los indicios constituirá la piedra filosofal en los ordenadores de la quinta generación: el PROLOG (PROGRAMMING in LOGic, programación lógica). Como en los lenguajes anteriores, el manejo de grandes fórmulas configuran su base, pero



desde una óptica diferente, la cual parte de la lógica matemática

## MAQUINAS PARLANTES

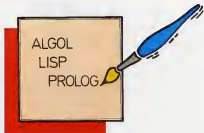
La síntesis y reconocimiento de voz por parte de un ordenador, ha sido uno de los campos más estudiados en los últimos años con objeto de conseguir establecer una relación más estrecha entre el hombre y la máquina. Hoy en día, escuchar la voz, más o menos metálica de un ordenador, es una realidad. Nuestro Spectrum no es una excepción y mediante un sencillo interface, el CURRAN  $\mu$ SPEECH, repite a través del televisor las frases que nosotros introducimos por el teclado, fuerte y claro.

El reconocimiento de voz es otro tema, debido a la distinta modulación y entonación que cada persona imprime en las palabras que pronuncia. Además, existe también el problema de la ambigüedad en el lenguaje, es decir, una misma palabra puede significar cosas diferentes según su contexto. Es un campo todavía en desarrollo, aunque algunos equipos ya realizan aplicaciones prácticas.

El MICRO COMMAND es un convertidor analógico digital, el cual se conecta en el slot de expansión del Spectrum. Mediante un proceso de aprendizaje en el cual repetimos ante un micrófono las palabras que el ordenador deseamos reconocerlo, éste acaba por hacernos caso, aunque su vocabulario está limitado a unas pocas palabras. Su mensaje quedó reservado exclusivamente al usuario que le enseñó, siendo necesario volver a reprogramarlo cada vez que el micrófono cambia de manos, pues de otra manera no reconocerá la nueva voz.

## VISION ARTIFICIAL Y ROBOTICA

Otra área todavía en desarrollo, es la dedicada a dotar a los ordenadores del sentido de la vista. Hasta ahora, el problema del reconocimiento de objetos está resuelto, siempre y cuando éstos



Lenguaje como el ALGOL, LISP y PROLOG, ya han sido diseñados para la escritura de programas inteligentes.

sean planos. Se utilizan cámaras de televisión enfocadas, por ejemplo, a las cintas transportadoras, las cuales en colaboración con robots, indican a éstos el momento en que deben recogerlos, ajustarlos en una máquina para su mecanización y posteriormente, tomarlos de allí y depositarlos en la cinta para su transporte a otra sección de la cadena.

La ficción, antesala de una realidad más o menos próxima, valida un futuro de fábricas controladas por enormes monstruos mecánicos, exentas de personal humano. «El poder de la máquina sobre el hombre». No obstante, la situación quizá no sea tan desalentadora, siempre y cuando se respeten leyes básicas como las enunciadas por Isaac Asimov en su ficticio manual de robótica:

- I. Un robot no debe dañar al ser humano, o por involuntariedad, permitir que sea dañado.
- II. Obedecerán disciplinadamente las órdenes dictadas por el hombre, siempre y cuando no vayan en contra del primer principio.
- III. Un robot debe proteger su propia existencia hasta el punto, en el cual, no se entre en conflicto con los principios anteriores.

El futuro forma ya parte del presente. INVESTRO-NICA, presentó durante el SIMO '84 celebrado en Madrid, un pequeño robot, el cual aceptaba las órdenes que recibía sin trukeos. Su utilidad es discutible, y no pasa de ser un mero entretenimiento, pero es una buena muestra de cómo los robots pueden invadir en breve nuestro entorno doméstico.



Los sistemas expertos han sido aplicados en el campo de la medicina, y en el de la seguridad de centrales nucleares.



La Quarta Generación es un proyecto japonés sobre inteligencia artificial, que cuenta con un presupuesto de 500 millones de dólares y una duración de diez años.





# LA PRESA



A fértil región de Rurilandia está amenazada por unos extraños artefactos voladores que colisionan contra los muros de sus grandes presas, ocasionando terribles destrozos en cada impacto. Sus habitantes han de actuar con rapidez y precisión si quieren evitar el desastro que se les avecina: la inundación de Rurilandia. Para ello, se ha situado al pie de la presa un potente cañón que deberá ser disparado con autómata, sangre fría y exactitud por algún voluntario.

¿Quieres ser tú ese avecedo voluntario? ¡Sí! Pues saca pecho y adelante. La defensa de Rurilandia es tuya.

## EL PROGRAMA

El juego consiste en impedir que los artefactos voladores que aparecen por la derecha de la pantalla, abran una vía de agua en el muro contenedor de nuestra presa.

Nuestros enemigos aparecerán a una altura determinada por el margen derecho de la pantalla. Esta altura es en cada uno de los intrusos aleatoria, por lo cual nunca tendremos una noción de la misma.

Por cada enemigo destruido, nuestro Spectrum nos dará una puntuación diferente, dependiendo de la altura a la que se encuentre en ese momento nuestro blanco. Estos puntos se irán acumulando sucesivamente en un marcador que el jugador tendrá a la vista en todo momento de la partida.

Por cada impacto que recibe la presa por parte de nuestros enemigos, surgirá en la fachada el impacto lógico del bestial embos, dando impresión de que la presa se desmorona.



Para salvar el programa utilizaremos SAVE "PRESA".



En ciertos momentos en un marcador que el jugador tendrá a la vista en todo momento.



Cada vez que destruyas aparecerán algunos débiles pero poder de reche!







La partida finalizará cuando el agua que contiene la presa quede libre e inunde una zona de pantalla predeterminada por el ordenador.

Debido a la gran cantidad de gráficos definidos que utiliza el programa, debemos tener muy en cuenta los caracteres subrayados que surjan en el listado.

El manejo del cañón de defensa de la ciudad es bien fácil: sólo tenemos que apretar la tecla 1 cuando consideramos que nuestro proyectil y el intruso se encuentran en rumbo de colisión. Sobre los desperfectos que estos «kamikazes» cau-

san en la presa, es interesante hacer notar que no van abriendo un boquete en la misma, sino que los impactos producen un desmoronamiento parcial del muro, lo cual en ocasiones puede ser una ventaja, pues los residuos de un impacto pueden pasar a tapar el desperfecto ocasionado por una colisión anterior.

La estética del programa se ve amparada por la profusión de gráficos definidos (18 en total), que como es habitual se encuentran representados en el listado por los caracteres correspondientes subrayados.



```

1 REM *****
2 PCP = (C) J.M.NAYRAL *
3 REM *****
4 LET CONT=0
5 POKE 23650,0
6 BRIGHT 0: PAPER 1: BORDER 1: INK 0: CLS
7 DATA 255,136,136,136,136,225,162,162,162
8 DATA 255,136,136,136,225,24,24,24,24
9 DATA 255,139,139,139,225,28,28,28,28
10 DATA 255,139,139,139,225,28,225,28,225
11 DATA 255,136,136,136,225,24,225,225
12 DATA 255,139,139,139,225,24,25,25
13 DATA 255,139,139,139,225,25,25,25
14 DATA 254,192,192,192,229,20,20,17
15 DATA 255,136,152,166,205,20,72,207
16 DATA 191,196,68,24,240,67,26,26
17 DATA 178,64,64,24,44,14,25,1
18 DATA 1,17,22,44,228,188,136,208
19 DATA 25,1,12,39,24,68,88,224
20 DATA 128,0,104,288,50,20,6,1
21 DATA 8,16,31,127,263,127,51,14
22 LET Q$="ACCCFGHILNPPQR"
23 FOR N=3 TO LEN Q$
24 FOR P=0 TO Y
25 READ A
26 POKE USR (GHIN)+N,A
27 NEXT N
28 NEXT N
29 LET A$="ABCDEFGHIJKLMN"
30 LET B$="PQRSTU"
31 LET C$="VWXYZ"
32 LET Q$="ABCDEFGHIJKLMN"
33 LET Q$="ABCDEFGHIJKLMNPPPPPPPPPPPPPPPP"
34 LET X=17
35 PRINT AT 2,3: PAPER 0: ONE 9:0
36 PRINT PAPER 0: BRIGHT 1: " "; BRIGHT 0: PAPER
61: INK 9:0
37 FOR N=3 TO 15
38 PRINT PAPER 0: BRIGHT 1: " "; PAPER 0: BRIGHT
0: INK 9:0
39 NEXT N
40 PRINT PAPER 0: INK 9:0
41 PRINT PAPER 0: INK 9:0
42 PRINT PAPER 0: BRIGHT 1: INK 2:0
43 BRIGHT 1: AT 20,11: "0"
44 PRINT AT 0,10: PRINTOS= "IDENT"
45 LET S=0
46 LET S=0
47 LET S=INT (RND*25)+3
48 LET B=31
49 LET S=1
50 IF B=3 THEN GO TO 5000
51 IF AT$(A,B) > 76 THEN PRINT INK 0: BRIGHT 1: AT
A,B: "0" : PRINT AT A,B+1: " " : LET X=A: LET Y=B: GO
500: PRINT LET CONT=CONT+18: LET X=16: LET S=0: GO
TO 390
52 IF AT$(A,B)=0 THEN GO TO 2900
53 PRINT ONE 2: BRIGHT 1: AT A,B: "2", BRIGHT 0: " "
54 IF S=1 THEN GO TO 480
55 GO TO 680
56 IF X=1 THEN LET S=0: PRINT AT X+1,11: " " : LET
X=1: GO TO 220
57 GO TO 2900
580 IF INKEY="1" THEN LET S=1: GO TO 3000
59 GO TO 300
5905 PRINT PAPER 0: AT A-1,0: "1"
5910 PRINT BRIGHT 1: PAPER 0: AT A-1,0: "N"
5915 DEEP -01,00: PRINT PAPER 0: AT A-1: "0"
5920 DEEP -01,05: PRINT PAPER 1: ONE 0: AT A,0: "0"
5940 GO TO 300
5990 IF AT$(X,11) > 76 THEN GO TO 3200
5910 PRINT INK 0: BRIGHT 1: AT X,11: "0"
5920 PRINT AT X+1,11: " "
5930 LET X=X-1
5940 GO TO 300
5990 PRINT INK 0: BRIGHT 0: AT X,11: "0"
5910 PRINT AT X+1,11: " "
5920 LET Y=Y+1: LET Y=Y+10
5930 GO SUB 7000
5940 LET CONT=CONT+18
5950 LET S=10
5960 LET X=10
5970 GO TO 300
5990 RUN PRESA ROT
5910 RUN FIN DE JEU
5930 PRINT PAPER 0: BRIGHT 1: AT A,0: " "
5935 LET F=0
5940 PAPER 0: BRIGHT 1,
5950 PRINT AT A,0: "1"
5960 PRINT AT A+1,0: "0"
5965 PRINT AT A+1,0: " "
5970 FOR N=0 TO 15
5980 PRINT AT N+1,0: "0"
5990 PRINT AT N,0: "1"
5995 DEEP -06,00: M/2
6000 NEXT N
6010 RUN SALIDA DE AGUA
6020 FOR N=3 TO 31
6030 PRINT PAPER 0: BRIGHT 1: OVER 1: AT F,N: " "
6040 NEXT N
6050 FOR N=0 TO 21
6060 PRINT PAPER 0: BRIGHT 1: OVER 1: AT N,31: " "
6070 NEXT N
6080 FOR N=25 TO A-2 STEP -1
6090 PRINT PAPER 0: OVER 1: BRIGHT 1: AT N,0: "
6100 NEXT N
6110 FOR N=3 TO A-3
6120 PRINT PAPER 1: OVER 1: INK 0: AT N,0: " "
6130 NEXT N
6140 BRIGHT 1: BORDER 5
6150 PRINT OVER 1: AT A,0: " PULSA UNA TECLA PARA COME
NZAR"
6160 IF INKEY=" " THEN DEEP -01,INT (RND*23)+28: GO
TO 5610
6170 RUN
6180 DEF EXPLOSIONES
7000 PRINT AT X,Y: " "
7020 FOR Q=1 TO 6
7030 PRINT OVER 1: BRIGHT 1: INK INT (RND*6)+2: AT X+
Y,Q: "0" : AT X+1,Y,Q: "0"
7015 DEEP -01,40
7020 NEXT Q
7030 PRINT AT X,Y: " "
7040 PRINT AT X+1,Y: " "
7050 LET CONT=CONT+28: A
7060 PRINT AT 0,10: CONT
7070 RETURN

```